Name____________________________

EET 2261 Lab #8
Seven-Segment Displays and Clock Speeds

1. Using a Single Seven-Segment Display

The Dragon12-Plus2 board has four common-cathode seven-segment displays. As explained on pages 24-26 of the Dragon12-Plus2 board's User Manual, the anodes of the displays are connected to the HCS12's Port B, while each display's cathode is connected to one bit of Port P. In particular, PTP0 is connected to the leftmost digit's cathode, PTP1 is connected to the next digit's cathode, PTP2 is connected to the next digit's cathode, and PTP3 is connected to the rightmost digit's cathode.

To display a digit on one of the displays, we must send the correct pattern of bits for that digit to Port B, and then ground the cathode of the desired display by making the correct bit of PTP low. For example, to display the digit **2** on the leftmost display, we send $5B (or %01011011) to Port B and we send a LOW to PTP0. The code below will do the job:

```
       ABSENTRY Entry
       INCLUDE 'derivative.inc'

       ORG   $2000
Entry: CLI              ;Enable interrupts
       LDS    #$4000    ;Stack
       LDAA   #$FF
       STAA   DDRB  ;Configure PORTB for output
       STAA   DDRP  ;Configure PTP for output

       LDAA  #$5B ;Pattern to show 2 on seven-segment
       STAA   PORTB
       LDAA  #%00001110  ;Ground PTP0 for leftmost digit
       STAA   PTP
       BRA    *     ;Stop

       END
```

1. Create a CodeWarrior project named **Lab08SingleDigit,** choosing **HCS12 Serial Monitor** on the New Project Wizard's first page.

2. Enter the program listed above. When you run it, you should find that the leftmost seven-segment displays shows a **2**, while the other displays are dark.

3. Modify the program to show a **5** on the ***rightmost*** seven-segment display, while the other displays are dark.

4. Show me your working program. ___________

2. Using Multiple Seven-Segment Displays

1. We can easily modify the previous program to show the same digit on all four seven-segment displays. Create a CodeWarrior project named **Lab08MutipleDigits**, choosing **HCS12 Serial Monitor** on the New Project Wizard's first page.

2. Copy and paste your code from above into the new project. Then modify it to display **AAAA** on all four seven-segment displays. *Hint: Since we're showing the same digit on all four displays, this program does not require you to use the multiplexing technique discussed in class. You simply need to send out the code for* ***A*** *and then enable all four displays.*

3. When you run the program, you may notice that the displays are dimmer than they were when you were showing a single digit. That's because seven-segment displays draw a fairly large current, and when all four of them are lit up they make a considerable demand on the Dragon board's power supply.

4. Show me your working program. ___________

5. Although it's easy to show the same digit on all four displays, showing different digits on different displays at the same time is trickier. Suppose we want to show the number **2014**. As you learned in class, we must do something like this:
    - Show a **2** on the leftmost display for an instant.
    - Then show a **0** on the second display for an instant.
    - Then show a **1** on the third display for an instant.
    - Then show a **4** on the rightmost display for an instant.
    - Then repeat, starting again with the leftmost digit.

   If we move from one display to the next one quickly enough, all four displays will appear to be lit up at the same time. Following this strategy, modify your program so that it shows **2014** on the seven-segment displays. To control how quickly you move from digit to digit, use your delay subroutine from previous programs, but adjust the delay time to about 1 millisecond.

6. Show me your working program. ___________

3. A Look-Up Table for the Seven-Segment Display

1. Create a CodeWarrior project named **Lab08SevenSegmentLookUpTable**, choosing **HCS12 Serial Monitor** on the first page of the New Project Wizard.

2. Write a program that reads the value of the rightmost four DIP switches on the Dragon12 board and then uses a look-up table to show this hex value (which will be between 0 and $F) on a seven-segment display. For example, if the rightmost four DIP switch settings are ON-OFF-ON-OFF (which is %1010), then the seven-segment display should show an **A**.
   Hints:
   - First you'll have to figure out which value to send to the seven-segment display to show each of the hex digits between 0 and F. For example, in the first program

above I told you that $5B is the value to show a ***2****. What is the value to show a* ***0****, to show a* ***1****, and so on?*
-Next, set up a look-up table containing these values.
-Your program should read the value on the DIP switches and then find the corresponding entry in the look-up table. This part of the program is similar to what you did in Lab05LookUpTable.

3. Show me your working program. __________

4. Counting on a Seven-Segment Display

1. Create a CodeWarrior project named **Lab08SevenSegmentCounter**, choosing **HCS12 Serial Monitor** on the first page of the New Project Wizard.

2. Using the look-up table that you created for the previous program, write a program that counts up from 0 to $F and displays the count on a seven-segment display. It should count slowly enough that you can read each digit. After reaching $F, the counter should recycle to 0 and start over again.

3. Show me your working program. __________

5. Using the HCS12's Phase-Locked Loop to Adjust Clock Speed

You've previously learned that the HCS12's bus clock normally runs at a frequency of 24 MHz. But as you learned in class, the clock frequency is programmable, using a feature on the chip called a phase-locked loop. Below you will re-program the phase-locked loop to cause the bus clock to run faster or slower than normal.

1. First, let's verify that the normal bus clock frequency really is 24 MHz. We'll do this by combining some calculations with an oscilloscope measurement. Assuming the bus clock's frequency of 24 MHz, what is the system's cycle time?

   ____

2. Consider the code listing below. If Ports B and J have previously been configured to enable the Dragon12's LEDs as output devices, this code will cause the LEDs to blink on and off very fast. In the code listing, there's a blank next to each instruction. Using the Instruction Set Summary, fill in these blanks with the number of cycles for each instruction. Then add these numbers to find the number of cycles in the loop.

```
Loop:   COMA        ___
        STAA PORTB  ___
        BRA Loop    ___
```

   Number of cycles in loop = __________

3. Using your answers from the two previous steps, how much time is required to execute the loop once?

   ____

4. Each pass through the loop will either turn the LEDs on or turn them off. So a complete ON-OFF cycle requires two passes through the loop. Therefore, how much time is required for a complete ON-OFF cycle of the LEDs? And therefore, what will be the frequency of the LED's blinking?

   Predicted Time = ________________ Predicted Frequency = ________________

5. Create a CodeWarrior project named **Lab08ClockSpeed**, choosing **HCS12 Serial Monitor** on the first page of the New Project Wizard. Write a program that contains, in addition to the loop shown above, any additional code needed to configure the Dragon12's ports correctly.

6. Run the program, and the LEDs on the Dragon12 board should light up dimly. They're blinking on and off far too quickly for your eye to see. Use an oscilloscope to measure the period and frequency of the waveform on any of the Port B pins.

   Measured Period = ________________ Measured Frequency = ________________

7. Your measurements should match the predictions that you made above.

8. Call me over to check your work. __________

9. Our little program above gives us a way to tell what the bus clock frequency is, by measuring the frequency of blinking when we run the program. In particular, we found that the frequency of blinking is one-twelfth the bus clock frequency. This one-twelfth ratio will always hold true for this program:

$$\frac{Frequency\ of\ blinking\ when\ we\ run\ test\ program}{Bus\ clock\ frequency} = \frac{1}{12}$$

10. Suppose, for example, that we run our test program on a chip with a different clock frequency, and we find that the LEDs blink at 1 MHz. Then what must this chip's bus clock frequency be?

    ____

11. Suppose we run our test program on another chip and we find that the LEDs blink at 200 kHz. Then what must this chip's bus clock frequency be?

    ____

You verified above that our chip's normal bus clock frequency is 24 MHz. However, *this is true only when the chip is being controlled by CodeWarrior. Otherwise, the normal bus clock frequency would be different.* In other words, the Dragon12 board's crystal oscillator gives a default bus clock frequency different from 24 MHz. But to communicate with CodeWarrior (using a program called the Serial Monitor, which is programmed into the HCS12's Flash memory), the bus clock frequency must be increased to 24 MHz. To see that this is true, we must free the system from CodeWarrior's control and run our program again. Here's how to do it:

1. Add the following two lines to the end of your program (but before the END directive). These lines define something called a reset vector, which we'll discuss next week. Put simply, these lines tell the chip the starting address of the program that it should run when the user presses the RESET button on the Dragon12 board.

   ```
   ORG $FFFE   ;Set up Reset vector.
   DC.W Entry
   ```

2. Run the program. The oscilloscope should display the same period and frequency as it did earlier, telling us that the system's bus clock frequency is still 24 MHz.

3. Press CodeWarrior's Halt button to stop the program. The oscilloscope will display a flat line since you've stopped the program.

4. To free the Dragon12 from CodeWarrior's control, locate the two DIP switches near the Dragon12's lower right corner. Notice the word LOAD under the leftmost switch, and the word RUN above it. Up to now, we've always had this switch in the LOAD position, which is where it needs to be to communicate with CodeWarrior. Move the leftmost switch to the RUN position (but leave the other switch alone). By doing this, you're telling the system to operate by itself, without CodeWarrior's control.

5. Press the Dragon12's RESET button. When you do this, CodeWarrior will complain that it has lost communication with the target system. But the program on the HCS12 will run anyway, and the oscilloscope will display a waveform with a new period and frequency. Record these values below.

   Measured Period = ________________ Measured Frequency = ________________

6. Therefore what is the chip's bus clock frequency when it's not being controlled by CodeWarrior?

   ____

7. Call me over to check your work. __________

8. To put the Dragon12 system back under CodeWarrior's control, move the DIP switch back to the LOAD position and press the Dragon12's RESET button. (You may also have to close CodeWarrior's debugger and re-open it.)

Now let's reprogram the bus clock frequency to a new value.

1. Create a CodeWarrior project named **Lab08PhaseLockedLoop**, choosing **HCS12 Serial Monitor** on the first page of the New Project Wizard.

2. Copy and paste your Lab08ClockSpeed program into this new project's main.asm file, and run it to make sure it works.

3. Using what you learned in class about the HCS12's phase-locked loop, modify the program so that it gives a bus clock frequency that is **one-half** of the bus clock frequency that you calculated on the previous page.

4. Download and run the program. (In this step, and from here onward, you should leave the Dragon12's DIP switches in the LOAD position.)

5. Use the oscilloscope to measure the period and frequency of the LED's blinking.

   Measured Period = ________________ Measured Frequency = ________________

6. Therefore what is the bus clock frequency?

   ____

7. Your answer to the previous question should be one-half of the last value you calculated on the previous page. Call me over to check your work.

   ____

8. By changing the values of SYNR and REFDV, we can get many different clock frequencies. In the table below, predict the bus clock frequency and the LED blink frequency for each pair of values for SYNR and REFDV. Then re-run your program with these different values for SYNR and REFDV, and measure the actual LED blink frequency.

SYNR	REFDV	Predicted Bus Clock Frequency	Predicted LED Blink Frequency	Measured LED Blink Frequency
0	0			
1	4			
2	1			
5	1			
4	0			

9. Call me over to check your work. __________